The Role of Knowledge and Certainty in Understanding for Dialogue

Susan L. Epstein[1,2], Rebecca J. Passonneau[3], Joshua Gordon[4], and Tiziana Ligorio[2]

[1]Hunter College and [2]The Graduate School of The City University of New York

[3]Center for Computational Learning Systems and [4]Departement of Computer Science, Columbia University

susan.epstein@hunter.cuny.edu, becky@cs.columbia.edu, joshua@cs.columbia.edu, tligorio@gc.cuny.edu

Abstract

As people engage in increasingly complex conversations with computers, the need for generality and flexibility in spoken dialogue systems becomes more apparent. This paper describes how three different spoken dialogue systems for the same task reason with knowledge and certainty as they seek to understand what people want. It advocates systems that exploit partial understanding, consider credibility, and are aware both of what they know and of their certainty that it matches their users' intent.

Introduction

In *human-machine* dialogue a person (the *user*) and a spoken dialogue system (the *SDS*) communicate with speech to address a common task. Although they seek to understand one another (i.e., to perceive each other's intent), human-machine dialogue is often fraught with frustration for the human and uncertainty for the system. Our thesis is that a proficient system requires knowledge about how to agree with its user on exactly which objects are under discussion and what is to be done with them. This paper reports on three SDSs that take different approaches to these challenges. The most promising employs a clearinghouse for knowledge about what the system knows, hypothesizes, and expects, along with an extensive variety of rationales that it learns how to use. This rich cognitive structure supports flexible reasoning and interaction during dialogue.

Understanding benefits from a shared context, knowledge that allows speakers to focus upon the same objects (*targets*) and ways to talk about them. Because human speech is ridden with *disfluencies* (e.g., filled pauses, repetitions and self-repairs), and because a description may not identify a unique object, people consistently assure one another about their understanding, including which targets are in their common ground. This behavior, known as *grounding*, uses vocal gestures (e.g., uh-huh), speech, and non-verbal cues to confirm mutual understanding (Clark and Schaefer, 1989). Confronted by partial or inaccurate information about a target, an SDS too may be able to use knowledge to collaborate on the common ground.

To understand its users, an SDS must advance well beyond *speech recognition* (translation from audio signal to text string). Dialogue for a complex task may include multiple subtasks and targets of different kinds. Moreover, if an SDS cedes to the user some of its control over the path dialogue may take (*mixed initiative*), the SDS must determine both the targets and how they relate to one another.

An SDS *misunderstands* when it misinterprets what it has heard, and binds some target or target feature incorrectly. Misunderstanding is common in human-machine dialogue, but difficult to detect and recover from. A *non-understanding* occurs when the SDS cannot go from an input audio signal to a useful representation of what has been said. A typical commercial system's response to a non-understanding is to ask the user to repeat. Repeated non-understandings can drive the system to end the dialogue.

To guard against misunderstanding, a commercial SDS often grounds *explicitly*: it repeats what it believes the user has said and insists upon confirmation before it proceeds. That drive for accuracy (e.g., "I heard you say 17Z946-AQ347R. Is that correct?") annoys many users. Moreover, to prevent the user from saying the unexpected, the SDS often maintains *system initiative*, that is, determines what is under discussion, and even what may be said.

In response to these challenges, we advocate novel approaches to partial information and certainty for SDSs. We recently introduced partial understanding as a confident interpretation of some part of the user's intent, one that engenders a question whose answer could support and enhance that interpretation (Gordon, Epstein and Passonneau, 2011). This paper elaborates on how partial understanding can avoid both non-understandings and misunderstandings. For example, given a person's full name as a target, with recognition confidence high for the last name but low for the first, a traditional SDS might re-prompt for the full name or signal non-understanding. Partial understanding would engage the user in a subdialogue that grounds the last name, and then elicits a first name consistent with the database and similar to the poorly-recognized first name.

We describe three SDSs for the same task: book orders at the Heiskell Braille and Talking Book Library, part of the New York Public Library and the Library of Congress. Heiskell's patrons order their books by telephone and receive them by mail. Book requests are by title or author, as often as by catalog number. All three SDSs have ample procedural knowledge about communication and dialogue. They know that speakers should take turns, and that listening provides a continuous audio signal, only some of which should be regarded as speech. They know that speech signals can be mapped to phonemes, and that only some sequences of phonemes are meaningful (*known words*). They know too that relevant word sequences provide possible bindings for targets or for *indicators* (e.g., yes, no).

To understand and respond to spoken input, commercial SDSs and applications where dialogue is subsidiary often rely on a *pipeline* architecture. A pipeline does best with well-recognized utterances about one class of objects over a limited vocabulary. In contrast, our library task is noteworthy for its confusability (e.g., the same name could be a patron, a title, or an author), its unusually long and complex responses (e.g., average title length of 6.4 words), and its scale: 5000 patrons plus a vocabulary of 54,448 words drawn from 71,166 books by 28,031 authors. Moreover, book titles are more like unrestricted language than like structured data, and more difficult to understand. This task also challenges automated speech recognition (*ASR*) with users' diverse native language, and with transmission noise and background noise in the audio signal. The next section describes the pipeline. Subsequent sections describe our three SDSs for the library task, explain how they differ in their use of knowledge and certainty, and focus on grounding. Finally, we discuss results and future work.

Knowledge and error in a pipeline

We take as a pipelined model *Olympus/RavenClaw*, an SDS architecture that has supported the development of more than a dozen spoken dialogue systems (Bohus and Rudnicky, 2009). When it detects voice activity in the incoming signal, such a system's audio manager labels and segments it into *frames* (short intervals of signal analysis output), and judges when the user began and stopped speaking (*endpointing*). It forwards the frames to an interaction manager (*IM*) that determines whether the user intends to continue speaking despite a pause. The IM also supervises while text strings from the speech recognizer are mapped to concepts by the subsequent natural language understanding (*NLU*) process. If the initial endpointing is not semantically coherent, the IM can override it (e.g., combine two speech segments into one utterance).

To transcribe the speech signal into text, the ASR relies on an *acoustic model* that maps speech sounds to phonemes, a *lexicon* that maps phoneme sequences to words, and a *language model* that indicates the probabilities of sequences of *n* words. The ASR forwards its output to a semantic parser, where a *concept* is an attribute of an object (e.g., a book's title). The parser tries to associate a given ASR text string with one or more concepts, and can skip words that cannot be parsed. A confidence annotator then selects at most one best parse.

The pipeline forwards that parse with its confidence to the SDS's dialogue manager. Given a confident parse, the *dialogue manager* performs one or more optional queries to backend databases, followed by a command to the natural language generator (*NLG*). For example, if the best parse indicated that the utterance was a book title, the dialogue manager could query the book database for similar titles, and then direct the NLG to formulate text to confirm the most similar (e.g., "Did you want *Jane Eyre*?"). The NLG forwards that text to the text-to-speech module, which in turn forwards the speech it generates to the interaction manager for transmission to the audio manager and then to the user. Given an unconfident parse or none at all, the dialogue manager invokes error handling appropriate to a misunderstanding or a non-understanding.

Errors may arise at many points in this pipeline. The audio manager might improperly endpoint the speech signal, and the IM might be unable to correct it. The user's words might not be in the system's lexicon. Disfluencies might disrupt the structure of spoken dialogue (Jurafsky and Martin, 2008). The ASR might mismatch signal to phoneme, or phoneme sequence to words. Finally, even with perfect speech recognition, a string might have a best (or first) parse that is incorrect, or even no parse at all.

Understanding in Three SDSs

We have built three systems for the library task. The first two are full SDSs that accept telephoned orders for up to four books, and can reference copies of Heiskell's entire book and (sanitized) patron databases. During a call, each dialogue addresses a sequence of subtasks: identify the user as a known patron, accept book requests, and offer an order summary. (Book requests are the most difficult.) Despite increasingly accurate ASR, deployed SDSs sometimes contend with word error rates (*WER*) as high as 68% (Raux et al., 2005). The work reported here has a similar WER to support research on strategies robust to poor ASR

The first SDS, *CheckItOut*, was developed with Olympus/RavenClaw modules. CheckItOut's decisions rely on *PocketSphinx*, a fast, efficient speech recognizer, and Wall Street Journal acoustic models. From a randomly-chosen subset of its database plus knowledge derived from its semantic grammar, CheckItOut generates both its lexicon and its language models with the *Logios* language compilation suite. Although its *Phoenix* parser is already robust to ASR noise, CheckItOut supplements it with productions derived from MICA dependency parses of book titles (Bangalore et al., 2009; Gordon and Passonneau, 2010). As a result, it can parse even poorly recognized title strings. Parses are scored by the *Helios* confidence annotator. The RavenClaw dialogue manager supplies task-independent error-handling mechanisms through a domain-dependent dialogue task tree (described below).

The second SDS, *CheckItOut+*, is identical to the first except for its dialogue manager, which models how people solve the problems that confront the system. It relies on information from all stages of spoken language understanding (*SLU*) to override the pipeline at three key decision points. While CheckItOut queries its database only when it understands (i.e., has a single confident parse whose slots match a known concept), CheckItOut+ queries with a full ASR text string (*voice search),* without recourse to Phoenix or Helios. When CheckItOut does not understand, it asks the user to repeat, but CheckItOut+ asks questions. (Further details appear in the next section.) Finally, while CheckItOut ends a call after several consecutive non-understandings, CheckItOut+ can *move on*, that is, ask the user to request another book and return to this one later.

The third SDS, *FX2*, implements some of the functionality of a full system with modules built from *FORR*, a cognitive architecture for learning and problem solving (Epstein, 1994). Rather than postulate a single decision rationale (e.g., voice-search confidence), a FORR-based system has *Advisors*, resource-bounded procedures that produce any number of *comments.* Each comment supports or opposes one action with a *strength* that reflects that Advisor's underlying rationale. (For example, comment strength may reflect the degree to which a match is the same length or sounds the same.) FX2 is built within *FORRSooth*, a new SDS architecture with six FORR-based *services*, each with its own set of heuristic dialogue Advisors. To make a decision, a service solicits comments about possible actions from its Advisors, tabulates a weighted combination of comment strengths that address each action, and identifies actions with high support. FX2 conducts selected subdialogues for the library task. FX2 also uses PocketSphinx but, like CheckItOut+, it is more resourceful in its responses to partial understanding and non-understanding. FX2 has a flexible dialogue representation, and a host of rationales with which to reason about what is expected, what it has hypothesized, and what has been said.

Noteworthy differences

An SDS should respond appropriately, effectively, and in real time to its user's speech. SDS performance is gauged not only by *success* (task achievement) and *cost* to the user (e.g., elapsed time), but also by user satisfaction, a non-trivial metric where faster and more accurate is not always better (Walker et al., 1997). All differences reported below are significant at the 95% confidence level under a *t*-test.

Before each call in the full SDS experiments reported here, the user retrieved a randomly-generated assignment from our website: a patron identity and data on four books. The user was told to request one book by author, one by title, one by catalogue number, and one by any method of her choice. (For a request by author, a query returns the three books by that author with the highest circulation.) Each experiment had 10 subjects make 50 calls each to the SDS. In the FX2 experiment, users interacted with the system by microphone rather than telephone, and interactions were subdialogues for a concept, such as author identity.

CheckItOut relies on matching

Even among many choices, people can ferret out an object that corresponds to a speaker's intent. Consider, for example, a book title the recognizer reported as SOONER SHEEP MOST DIE. Our *pilot study* gave similarly noisy ASR for 50 book titles, a plain text file of the library's 71,166 titles, and unlimited time offline, to each of 3 subjects (Passonneau et al., 2009). They correctly matched 74% of the ASR strings to a listed title.

CheckItOut matches such noisy ASR against its database with the Ratcliff/Obershelp similarity metric between two strings (*R/O score)*: the ratio of the number of correct characters to the number of characters (Ratcliff and Metzener, 1988). (For example, the R/O score for *Robert Lowell* and ROLL DWELL is 0.61.) CheckItOut's best matches for SOONER SHEEP MOST DIE are *Soon She Must Die, Why Someone Had to Die,* and *The Messenger Must Die.* Clearly, the first is the intended book. Indeed, given a single confident parse from noisy ASR for book titles (with a WER of about 70%), the search return with the top R/O score is the correct match about 65% of the time (Ligorio et al., 2010). A skilled human, however, can achieve 85.5% accuracy at this task, as we shall see shortly.

When CheckItOut produces a single confident parse for a title or an author, its dialogue manager searches for it in the database with the parsed ASR words. It then offers the user the return with the top R/O score, as in Figure 1. Although 65% accuracy is not satisfactory to people, another 6% of the time the correct match is elsewhere in the top 10 returns (Ligorio et al., 2010). For these cases CheckItOut+ and FX2 bring to bear additional knowledge already within the system, but with very different approaches.

CheckItOut+ tries harder

CheckItOut+ processes a single confident parse the way CheckItOut does. In the absence of a single confident parse, when CheckItOut would have signaled non-understanding, CheckItOut+ uses three learned models to advance the dialogue. If it has at least one ASR text string but no confident parse, *Model S* (for search) decides whether the most confident ASR text string is good enough to use in voice search. If so, CheckItOut+ searches three times, because the utterance might refer to a book by title, author, or catalogue number. *Model O* (for offer) determines whether to offer the return with the highest R/O score, or to request additional information from the user. Otherwise, if CheckItOut+ did not do voice search, it uses its *Model Q* (for question) to decide whether to question the user about another way to identify the book or merely to signal non-understanding. Figure 2(a) is an example of voice search; Figure 2(b) is a request for further description.

CheckItOut+'s models are the result of logistic regression on 163 features available at runtime. The models are based on data from an elaborate experiment that replaced CheckItOut's dialogue manager with an *ablated wizard,* a person given the same input and query facility as the SDS, and restricted to a limited set of dialogue acts. The models were learned from system features and wizard actions logged during 913 dialogues (Ligorio, 2011).

The knowledge embedded in these models supports a different kind of dialogue. The only subject who had called both CheckItOut and then, months later, called CheckItOut+, commented on the change: "This new system just doesn't let up on you." CheckItOut+ is indeed persistent, resulting in dialogue like Figure 2(b).

CheckItOut+'s models use 24 features in all, only 2 of which (author and title queries) are library-specific. An *adjacency pair* is the portion of the dialogue from one system prompt to just before the next one; it may include multiple user utterances. Model S determines whether to search on the ASR; its features describe the current book request (number of adjacency pairs, number of database or title queries for it), the dialogue (number of questions thus far), the context (whether this adjacency pair was initiated by an explicit confirmation, whether or not a non-understanding had just occurred), the SLU (average ASR word confidence, number of words covered by the best parse, number of parses for this ASR, whether this is the top grammar slot in the best parse tree), and the number of author queries in this request. Model O decides whether to offer a book, based on whether or not this is a new request, the total number of move ons, mean acoustic score, and mean and standard deviation of the R/O scores. Finally, Model Q decides whether to ask the user a question given the number of user utterances in this adjacency pair and features that describe the current request: number of author queries for it, number of partial explicit confirmations for it, maximum ASR word confidence, overall acoustic model score, whether there was a title slot in the best parse, the number of words not covered by the best parse, and the confidence in the best parse. Model Q also shares three features with Model S: whether the current adjacency pair was initiated by an explicit confirmation, average ASR word confidence, and number of author queries in this request.

CheckItOut: Sorry, I misunderstood. Can you repeat the author?
ASR: DIDN'T I AM THE .GOLDEN. ARM
CheckItOut: Is the title *The Man with the Golden Arm*?
ASR: NO
CheckItOut: Sorry, I misunderstood. Can you repeat the title please?
ASR: .A. .AMBLING. .THE. .GAME. .EDELMAN. STORY
CheckItOut: Is the title *Up and Running: The Jami Goldman Story*?
ASR: YES

Figure 1: Part of a CheckItOut dialogue. ASR is in uppercase.

CheckItOut+ improved task success. Throughput rose — the number of ordered books increased from 3.22 with CheckItOut to 4.00 per call, while the elapsed time per ordered book decreased from 65.57 to 56.01 seconds. Costs rose too — the system spoke more, and the user had to speak more often. Total elapsed time per call rose from 210.93 to 223.96 seconds, while the elapsed time per correct book decreased from 87.89 to 82.95. CheckItOut+ identified more books correctly on every call (2.70 instead of 2.40), but it also got more wrong, which forced the user to correct it more often.

To gauge user satisfaction, each subject completed the same questionnaire about her experience with CheckItOut or CheckItOut+ 3 times in the course of her 50 calls. Although their answers were consistent (Cronbach's $\alpha = .97$),

(a) **CheckItOut+**: What's the next book?
ASR: .FOR. .NONDRIVERS. .SCHULLER. CHAPMAN
CheckItOut+: Is the author Fern Schumer Chapman?
ASR: YES
CheckItOut+: Would you like *Motherland beyond the Holocaust: A Mother-Daughter Journey to Reclaim the Past*?
ASR: YES

(b) CheckItOut+: What's the next book?
ASR: DUST .TO. DUNE
CheckItOut+: Is the title *Dust to Dust* ?
ASR: NO
CheckItOut+: Sorry, I misunderstood. Can you repeat the title please?
ASR: GOING THE .OTHER. ARETHA THAT
CheckItOut+: Can you please give me the catalogue number?
ASR: THREE ONE NINE NINE EIGHT
CheckItOut+: *Gorbachev: Heretic of the Kremlin.* Got it!

Figure 2: CheckItOut+ dialogue fragments harness partial understanding despite unconfident ASR (delimited by periods). (a) Use of voice search. (b) Request for further description.

there was only one significant difference: CheckItOut+ users more often indicated that they had to pay close attention while using that system, probably because this SDS responds to partial understanding with a question, not a request to repeat. There was no statistically significant support, however, for user preference of one system over the other, even though CheckItOut+ identified more books correctly and processed individual book requests faster.

FX2 constructs a dynamic representation

As required by RavenClaw, CheckItOut's dialogue manager is a *task tree*, a hierarchy of pre-specified dialogue procedures (e.g., *Login,* in Figure 3(a)). Some leaf nodes (e.g., *Get area code*) issue prompts to determine values for concepts. The task tree is executed depth-first, but preconditions on nodes can redirect it. For example, *Inform lookup error* will return control to *Login* if there is no match on the telephone number. The task tree effectively preprograms dialogue flow. (RavenClaw's support for limited mixed initiative was not used here.) CheckItOut+ uses the CheckItOut task tree if there is a confident parse, and otherwise relies on its three models.

Instead of a static task tree, in a FORRSooth SDS the SATISFACTION service maintains an *agreement graph*, a dynamic structure that represents what is under discussion. An *agreement* is a subdialogue to bind a target (e.g., the first book in an order). Initially, an agreement graph node represents a target or an attribute of a target as its child. An example for author appears in Figure 3(b). Each node also

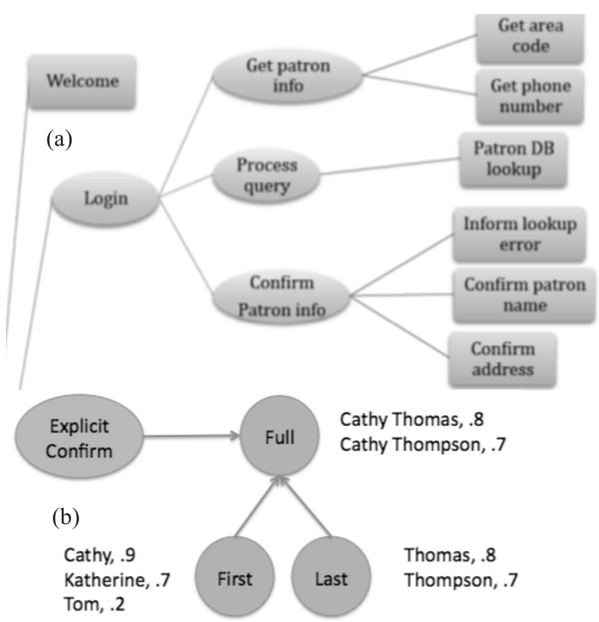

Figure 3: (a) Part of the task tree for CheckItOut. (b) A fragment of an FX2 agreement graph for author name. First and last are attribute children; hypotheses appear with their respective merits. The node on the left represents a decision to ground.

records progress toward its grounding, as described below. The graph retains partial understandings (e.g., a patron's perfectly recognized first name) between user utterances.

FORRSooth's INTERPRETATION service constructs *hypotheses* (the system's beliefs in possible values) for agreement nodes. Each of its nine matching Advisors represents a different way to formulate a hypothesis about what the user has said (Gordon, Passonneau and Epstein, 2011). Resources for them include Olympus modules, and DoubleMetaphone representations of titles and authors. One matching Advisor simulates CheckItOut; it proposes top returns from a query based on Helios' most confident parse. Two Advisors do not search at all; they propose hypotheses based on the parse of either all confident words, or all words, in the ASR. Another does voice search based on concepts identified by all parses, and re-ranks the returns with strength proportional to the number of confident words. Two pairs of Advisors use R/O score with voice search in the database, against either the target values or their DoubleMetaphone representations. One pair does voice search in the title, author, and catalogue-number tables; the other pair searches them with the concatenated terminals from the Phoenix parses. Finally, one uses parses both to decide whether the request was a title, an author, or a catalogue number, and to rank the search returns.

Comment strengths for voice-search Advisors are computed from such metrics as R/O score, ASR word-level confidence, and relative position and edit distance between the ASR and a search return. Comment strengths for parse-oriented Advisors are based on overall and word-level confidence, and the number of words not covered by the parse. The *merit* of a hypothesis gauges the extent to which Advisors' comments support it over alternatives for that node. Merit equals the percentile into which the (normalized) strengths of the comments that support a hypothesis fall, relative to others for the same node. Based on matching Advisors' comments, SATISFACTION records hypotheses with their merits on the corresponding agreement nodes.

INTERPRETATION also has five merging Advisors that revise merit or formulate new hypotheses for a node from existing hypotheses for the partial information in its attribute children. Based on R/O scores, two Advisors (one for patrons and one for authors) propose full name hypotheses based on existing hypotheses for the first and last names. Two others formulate hypotheses from fragments of titles or telephone numbers with high R/O scores. The fifth merging Advisor revises the merits of existing hypotheses based on grounding status, described below. As with matching, voting determines a single action to take. FX2's INTERPRETATION service produces relatively reliable hypotheses for patron names; their quality degrades gracefully as ASR performance declines (Gordon, Passonneau and Epstein, 2011).

Grounding supports understanding

In both CheckItOut and CheckItOut+, grounding behavior is managed by Ravenclaw's error-handling routines. When Helios identifies a sufficiently confident parse, those systems search on it, and then offer the best query result to the user. Both SDSs also retain at most one binding for a target between user turns. In contrast, a FORRSooth-based SDS entertains multiple hypotheses for agreement nodes, and retains some hypotheses until a node is bound.

FORRSooth's GROUNDING service monitors the merit values on the agreement graph, relying on learned Advisor weights (Gordon, Epstein and Passonneau, 2011). It proposes node values to the user, and elicits corroboration, further information, or tacit agreement when it determines that a hypothesis requires no confirmation from the user (e.g., "Got it. Next book?"). These 23 Advisors reference such additional information as ASR word confidence, how long the dialogue has progressed, and whether a hypothesis is for a target node or its child. They seek to advance the dialogue with fewer questions and little chance of error. For example, when a hypothesis for the last name agreement has very high merit but conflicts with an existing hypothesis for the first name of the same target, FX2 detects the conflict and considers both hypotheses unconfident.

If GROUNDING cannot bind an existing hypothesis to its target as a value, it considers how to discuss its uncertainty with the user. A *grounding agreement* either elicits confirmation for a particular hypothesis or seeks to disambiguate between competing ones. A grounding agreement is attached to a target or to an attribute node whose hypothesis it addresses. It has an *expectation* for its anticipated user response, and it specifies a grounding action. Grounding actions in FORRSooth include explicit confirmation (e.g., "Is the title *Dust to Dust*?"), implicit confirmation (e.g., "By John Wooden"), and disambiguation (e.g., "Was that 'Jane' or 'John'?"). Based on hypotheses, their merits, and expectations associated with grounding agreements, 23 additional FX2 GROUNDING Advisors determine when and where to append a grounding agreement to the graph. They consider, for example, whether there are competing hypotheses for the same target, if a hypothesis is for a title or a subtitle, and if it is the first attempt to ground this node's value. In Figure 4, FX2 finds two promising but competing hypotheses for the same target, and offers them to the user. Although the response matches the grounding agreement's expectation, INTERPRETATION fails to understand the response, and FX2 tries a different grounding action.

FORRSooth is intended to learn rapidly to tailor an SDS's responses to its task. The weights a FORR-based system uses to select an action are learned from labeled training examples. A FORRSooth-based SDS learns one set of weights for each service and each concept. We used FORR's domain-independent *RSWL* (Relative Support Weight Learning) algorithm to learn weights for FX2's GROUNDING and INTERPRETATION Advisors. *Relative support* for an action is the normalized difference between the comment strength with which an Advisor supports an action and the strength with which it supports other actions. RSWL reinforces Advisors' weights in proportion to their comment strengths (Petrovic and Epstein, 2007).

In FORRSooth, learning is currently offline, with training examples gleaned from the system's independent dialogues, and labels from user-supplied indicators or from wizard experiments. Because GROUNDING is essential to exploit partial understandings, FORRSooth provides a mechanism to extract preliminary training examples. In *wizard mode,* while a user speaks with the SDS, a GUI provides a human wizard with the SDS's current hypotheses and merits for all targets. Each time the wizard chooses whether to ground a target, and which grounding action to use on it, a training example is logged.

FX2 begins with some prior knowledge about how to use its Advisors well. For the *baseline version* we learned preliminary INTERPRETATION weights on prerecorded *development data*. (A training example was a target with its ASR and correct binding.) We gleaned training examples for GROUNDING in wizard mode while 4 subjects made 30 calls to FX2, with hypotheses and merits produced under those weights. Finally, we retrained FX2 on the development data and the new wizard-mode training examples extracted for both INTERPRETATION and GROUNDING, several hundred examples in all (Gordon, Epstein and Passonneau, 2011). The resultant *learned version* of FX2 produced the dialogue in Figure 4. The learned version virtually eliminates misunderstandings; it prefers to ground by disambiguation (52%) and repetition (32%), with occasional recourse to confirmation (15%) and other strategies (1%).

Four subjects had 30 subdialogues with the baseline version and 30 with the learned version. We evaluated the two versions on subdialogues to bind either a book title or a patron name. The *length* of a subdialogue is measured here by the number of times the system spoke. Length improved from 4.36 in the baseline to 3.86 after learning. (In comparison, CheckItOut+ averaged length 6 for book titles.) If a subdialogue identifies the requested object it is *correct*. If

FX2: What title would you like?
User: *Family and Friends*
ASR: FAMILY .FRIENDS.
FX2: I have two guesses. The first is *Family and Friends.* The second is *Family Happiness.* Is it either of these?
User: The first one
ASR: .NEXT. .FIRST.
FX2: Let's try something else. Is the full title *Family and Friends?*
User: Yes
ASR: YES

Figure 4: Grounding behavior in an FX2 dialogue fragment, with what the user actually said as well as what the system "heard."

it identifies some object it is *complete*; otherwise it is *incomplete*. *Precision* is the ratio of correct to completed subdialogues. Despite an estimated WER of 66%, precision rose with learning, from 0.65 in the baseline to a perfect 1.00 (n = 120) in the learned version. *Recall* is the ratio of correct subdialogues to correct plus incomplete subdialogues. Recall dropped somewhat with learning; it went from 0.78 in the baseline to 0.71. Finally, F is the harmonic mean of precision and recall. F rose with learning, from 0.72 in the baseline to 0.83 in the learned version.

Discussion

Our three SDSs all rely on the same speech recognizer and databases, but FX2 uses them with considerably more success. CheckItOut+ monitors its pipeline, and behaves differently when it believes an error has arisen. FX2 employs a variety of rationales observed in human behavior during our pilot study and our wizard experiment, and learns to balance them, instead of pre-specifying their interaction.

Two of our systems take novel approaches to the role of certainty in SDS decision making. CheckItOut+ includes system-component confidence values and other metrics on performance accuracy (e.g., number of questions) to select its actions. Its models recognize when CheckItOut+ has a partial understanding, when it has a reasonable guess, and when it should seek another way to identify a target. These models are procedural metaknowledge learned from features for components where SDS developers know that errors are likely to arise. In contrast, FX2 scales certainty as merit, and represents partial information explicitly. It links targets in its agreement graph with plausible values, and formulates grounding behaviors for strong hypotheses. The agreement graph is a clearinghouse for commentary on what may or may not have been intended by the user, as construed by FX2's Advisors. In this way, FX2 harnesses partial understanding and multiple perspectives to match spoken input and domain knowledge to targets.

Some of this work has appeared in venues for natural language processing, human cognition, or system design. Here, we have sought to compare and analyze it, primarily to clarify the role of knowledge and certainty in understanding during dialogue. Task-specific knowledge about objects often provides contextual data against which to match accurate input. Two SDSs here, however, use contextual data to generate plausible hypotheses from imperfect input. One learns models of human decision making from thousands of instances. The other learns to combine many rationales that were effectively gleaned from a few hundred instances of human behavior. FX2's rationales propose hypotheses, gauge their accuracy, and may confirm them with the user. They are knowledge about how to match and how to work toward common ground.

The agreement graph also represents the conversational state, that is, what dialogue utterances have contributed to the current common ground with respect to task objects. FX2 allows a new utterance to change an agreement graph for a target already addressed by an earlier utterance. It periodically removes weak hypotheses, and makes decisions based on the merits of those that remain.

Much of this work is task-independent, including merit, RSWL, and the agreement graph. Indeed, 52 of FX2's 60 INTERPRETATION and GROUNDING Advisors are provided by FORRSooth. The other eight, intended only for names, apply important ideas about the way attributes identify an object uniquely. Current work generalizes them for other concepts and other identifiers. Our best wizards' problem-solving behaviors are also task-independent (and likely to pertain to other cognitive systems as well): search before you reply, disambiguate among likely search returns, and notice when no match looks reasonable.

Our experiments made clear that people want an SDS that is not only fast and effective, but also transparent and easy to converse with. Users also need confirmation, so that they know what the system believes. For example, even when a wizard was both certain and correct, several users complained that they were surprised at the end of the call to hear that the order summary they had demanded actually included the correct books. FX2's fine-grained grounding provides more transparency than many SDSs about how the common ground evolves.

FORRSooth extends FORR with parallel computation and the ability to propose hypotheses, but it remains a work in progress. FX2 is its first application, and some of its services (an INTERACTION manager, GENERATION of natural language, and DISCOURSE to focus of attention and manage objects) are not yet implemented. SATISFACTION requires further development.

Human expertise inspires and supports FORRSooth in a variety of ways. To create Advisors and devise strengths for their comments, we continue to mine both commentaries from subjects in the pilot study, and the features that drive CheckItOut+'s models. A FORR-based system traditionally uses a three-tiered hierarchy of Advisors; some are always correct, and others heuristically formulate behavior sequences. Both kinds are a focus of current work for every service. Subjects' comments have also led to some Advisors that oppose actions (e.g., do not ground) as well as others that support them. There is even an INTERPRETATION Advisor that simulates an expert CheckItOut wizard.

Other cognitive architectures have also begun to address dialogue. CogX retains the pipeline; its perceptron learns to discriminate among the many more parses its relaxed grammar rules produce (Lison and Kruijff, 2009). Current work in SOAR uses written subdialogues to teach an apprentice goal-oriented plans using an extendible, but thus far small, vocabulary (Assanie and Laird, 2011).

A machine's context, however, is not a human one. We do not restrict FORRSooth to reasoning mechanisms and behaviors evidenced by people. After all, an SDS does not have the world and social knowledge that people do. FORRSooth's Advisors also capture the perspective of the system. For example, one INTERPRETATION Advisor, before any database query, relies on a learned classifier to remove from the ASR tokens likely to correspond to noise. Whether or not people do this, an SDS certainly should.

Conclusion

As we demand more of them, SDSs will find it increasingly difficult to understand their users. Future SDSs will have to detect and address subtasks, and consider how speech about attributes of objects can be exploited to identify those objects with certainty. People, meanwhile, will continue to expect the efficient, virtually error-free performance traditional SDSs now produce when they receive short utterances from a limited vocabulary.

CheckItOut+ models, to some extent, how people make the kinds of decisions an SDS must make. Some of its models' features reference dialogue history, but it retains no partial information from one adjacency pair to the next. Making decisions like a person proves to be less effective than FX2's ability to collaborate with the user on the common ground, and thereby minimize misunderstandings. Nonetheless, the features behind human decisions are a rich, task-independent resource for dialogue decision rationales, one that FORRSooth exploits to its advantage.

FX2's agreement graph is a dynamic representation of what it believes the user meant across multiple utterances, and its certainty in that information. It begins as a model of the task (a set of targets to be bound), but rapidly becomes a representation of what the system suspects, what it has confirmed, and what remains to be determined. The agreement graph makes it possible to tell the user what the system "thinks" (as in Figure 4), and FORRSooth's Advisors can explain why it thinks so (e.g., "this sounds like the first name and is similar to the last name"). FORRSooth's services and most FX2 Advisors are task-independent procedures that capture a broad range of reasons to consider something a good match or worthy of consideration for binding. Together they use knowledge and certainty to support understanding with precision as good, or better than, the best of our human wizards.

Acknowledgements

The National Science Foundation supported this work under awards IIS-084966, IIS-0745369, and IIS-0744904.

References

Assanie, M. and J. Laird. 2011. http://www.eecs.umich.edu/~soar/sitemaker/workshop/19/assanie-FinalWorkshop.pdf.

Bangalore, S., P. Bouillier, A. Nasr, O. Rambow and B. Sagot 2009. MICA: a probabilistic dependency parser based on tree insertion grammars. Application Note. Human Language Technology and NAACL. Boulder, CO: 185-188.

Bohus, D. and A. I. Rudnicky 2009. The RavenClaw dialog management framework: Architecture and systems. Computer Speech and Language 23(3): 332-361.

Clark, H. H. and E. F. Schaefer 1989. Contributing to discourse. *Cognitive Science* 13: 259-294.

Epstein, S. L. 1994. For the Right Reasons: The FORR Architecture for Learning in a Skill Domain. *Cognitive Science* 18(3): 479-511.

Gordon, J., S. L. Epstein and R. Passonneau 2011. Learning to Balance Grounding Rationales for Dialogue Systems. In Proceedings of SIGDIAL 2011.

Gordon, J. and R. Passonneau 2010. An Evaluation framework for Natural Language Understanding in Spoken Dialogue Systems. In Proceedings of Seventh International Conference on International Language Resources and Evaluation (LREC '10), European Language Resources Association.

Gordon, J., R. Passonneau and S. L. Epstein 2011. Helping Agents Help Their Users Despite Imperfect Speech Recognition. In Proceedings of AAAI Symposium Help Me Help You: Bridging the Gaps in Human-Agent Collaboration. AAAI Press.

Jurafsky, D. and J. H. Martin 2008. *Speech and Language Processing*, 2nd edition. New Brunswick, NJ. Prentice Hall.

Ligorio, T. 2011. Feature Selection for Error Detection and Recovery in Spoken Dialogue Systems. Ph.D. diss., Department of Computer Science, The Graduate Center of The City University of New York, New York, NY. Ph.D. thesis.

Ligorio, T., S. L. Epstein, R. Passonneau and J. Gordon 2010. What You Did and Didn't Mean: Noise, Context, and Human Skill. In Proceedings of Cognitive Science – 2010.

Lison, P. and G.-J. M. Kruijff 2009. Robust processing of situated spoken dialogue. 3In Proceedings of 2nd German Conference on Artificial Intelligence (KI '2009).

Passonneau, R. J., S. L. Epstein, J. B. Gordon and T. Ligorio 2009. Seeing What You Said: How Wizards Use Voice Search Results. In Proceedings of IJCAI-09 Workshop on Knowledge and Reasoning in Practical Dialogue Systems, Pasadena, CA, AAAI Press.

Petrovic, S. and S. L. Epstein 2007. Random Subsets Support Learning a Mixture of Heuristics. In Proceedings of FLAIRS 2007, Key West, AAAI.

Ratcliff, J. W. and D. Metzener 1988. Pattern Matching: The Gestalt Approach, Dr. Dobb's Journal.

Raux, A., B. Langner, A. Black and M. Eskenazi 2005. Let's Go Public! Taking a spoken dialog system to the real world. In Proceedings of Interspeech 2005(Eurospeech), Lisbon, Portugal.

Walker, M. A., D. Litman, J., C. A. Kamm and A. Abella 1997. PARADISE: A Framework for Evaluating Spoken Dialogue Agents . In Proceedings of Thirty Fifth Annual Meeting of the Association for Computational Linguistics (ACL), 271-280.